UNA MIRADA A LA COMPLEJIDAD COMPUTACIONAL Y SEGURIDAD EN LA PRÁCTICA DE LOS ALGORITMOS MD5 Y DES.

MIGUEL ÁNGEL MENDOZA GÓMEZ

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA

2017

UNA MIRADA A LA COMPLEJIDAD COMPUTACIONAL Y SEGURIDAD EN LA PRÁCTICA DE LOS ALGORITMOS MD5 Y DES.

MIGUEL ÁNGEL MENDOZA GÓMEZ

Trabajo de grado presentado como requisito para optar al título de Ingeniero de Sistemas y Computación.

Director

Ana María López Echeverry

Magister en ingeniería

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN

PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA

2016

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del jurado

_____

Firma del jurado

Pereira, 22 de noviembre de 2017

DEDICATORIA

Dedicado a toda mi familia que ha sido un gran apoyo en este trayecto lleno de tropiezos y aprendizajes.

AGRADECIMIENTOS

Se agradece al grupo de investigación Nyquist por su gran apoyo y guía en la realización del proyecto, especialmente a Ana María López Echeverry y Miller Ramírez Giraldo.

TABLA DE CONTENIDO

pág.

Lista de ilustraciones

Lista de tablas

Lista de ecuaciones

RESUMEN

La autenticación mediante el uso de contraseñas es una práctica común en los sistemas informáticos. Es una de las formas para que los usuarios tengan ciertos privilegios o acceso a determinada información. Las contraseñas protegen estos sistemas para que los usuarios no sean suplantados. Se plantea el caso en el que un atacante obtiene las contraseñas cifradas de un sistema informático e intenta obtener las contraseñas en texto plano para realizar procesos de autenticación en este sistema informático o en otro, cuando el usuario utiliza la contraseña para el acceso a diferentes servicios. Se realiza una comprobación de la dificultad en cuanto a la complejidad computacional para descifrar las contraseñas de los usuarios suponiendo que el atacante realizara pruebas de ataque por fuerza bruta y de diccionario, que son unos de los métodos de cracking más conocidos.

Para la comprobación de esta complejidad se hizo uso de una red donde se corrieron diferentes pruebas del software John The Ripper de manera distribuida, para medir el tiempo que toma descifrar diferentes contraseñas que fueron cifradas con los algoritmos DES y MD5. En estas mediciones se tomaron en cuenta dos tipos de contraseñas: contraseñas seguramente generadas y palabras comunes del idioma inglés.

Posteriormente se realizó un análisis de la disposición de los datos, una comparación entre la tendencia en gráficas de los tiempos medidos y la tendencia esperada según la complejidad computacional. Se llega a la conclusión sobre la tendencia exponencial de la complejidad de cracking respecto a la longitud de la contraseña, la recomendación de hacer uso de algoritmos de cifrado mejores que el DES y la recomendación del uso de salt por parte del sistema para asegurar la generación de contraseñas seguras independientemente de la suministrada por el usuario.

ABSTRACT

The authentication trough passwords is a common practice in computer systems. It is a way in what the users can have some privileges or access to specific information. Passwords protect these systems against an impersonation. It is posed the case in which an attacker obtains the text of cipher passwords of a computer system and tries to get the plain text of the passwords to do authentication processes in this system or another in which the user uses the same password. A check of computational complexity to decipher a password is made supposing that the attacker will try to do a brute force-attack or a dictionary attack that are the most common cracking methods.

To check this complexity it was used a red in which were ran different tests of distributed software John The Ripper to measure the time that takes crack different passwords that were encrypted with the algorithms DES and MD5. In this measurements were taken into account two kind of passwords: secure passwords and weak passwords.

Later was made an analysis of the data inclination. It was made a comparison between the graphics tendency of measured times and the expected tendency according to the computational complexity. It concludes about the exponential tendency against password´s length, it is recommended use better cipher algorithms than DES and that the system use salts to be sure that the password that generates the cipher text is a secure password no matter the password that the user uses.

INTRODUCCIÓN

El uso de las tecnologías de la información y comunicaciones es cada vez más extenso [1][2]. Este crecimiento viene acompañado de un aumento en la inversión para la seguridad informática debido a que los ciberataques pueden llegar a tener consecuencias muy graves en las empresas. [3]

Una práctica muy común en los sistemas informáticos es el uso de un inicio de sesión a través de un usuario y contraseña. Esta práctica se apoya en una rama de la seguridad informática llamada criptografía, la cual nos brinda herramientas de integridad, protección y confidencialidad. Una de estas herramientas es la función *hash*, que permite mapear un *message digest* con una simple entrada. Esto es ampliamente usado para la autenticación de contraseñas, ya que puede ser muy riesgoso guardar estas en texto plano. Lo que se hace es mapear la contraseña creada y guardar el *hash* generado por la función. Para autenticar un usuario se aplica la función *hash* a la contraseña ingresada y se compara con la contraseña guardada anteriormente. En caso de perder u olvidar la contraseña el usuario no podrá recuperarla, y tendrá que recurrir a un cambio de contraseña.[4]

Se plantea entonces el escenario donde un atacante adquiere los *message digiest* o valores *hash* generados, donde lo ideal sería que aun con esta información los atacantes no puedan saber cuál fue la contraseña que generó ese valor de hash. Esto hace que sea importante que los algoritmos implementados para la generación de estos valores sean lo suficientemente sólidos contra diferentes tipos de ataques y que cumplan ciertas propiedades de seguridad.

1. DEFINICIÓN DEL PROBLEMA

1.1 ANTECEDENTES DEL PROBLEMA

Ha habido diferentes vulneraciones de seguridad en donde se ha realizado cracking a contraseñas en diferentes sistemas de información. Algunos de los casos más relevantes serán expuestos a continuación:

En un incidente reportado por el CERT un conjunto de archivos de contraseñas fueron encontrados en un sistema comprometido.

En total el intruso apareció con una lista de 186126 cuentas y contraseñas cifradas. En el momento en que la colección de contraseñas fue descubierta el intruso había adivinado exitosamente 47642 de esas contraseñas a través del uso de una herramienta para descifrar contraseñas.

Dado que la mayoría de las entradas no provienen del sitio donde se encontró la colección parece que estas fueron tomadas de otro sitio por el intruso mientras que algunos de los archivos incluían información identificando el sitio de donde se había originado el archivo, cerca de esas 160000 entradas de los archivos contenían solamente usuario y contraseña cifrada. Fue reportado que el intruso tenía una colección de contraseñas que había obtenido gracias al uso de una aplicación de software para capturar la información que circula en una red. Esto nos muestra que a pesar de que las organizaciones tengan implementados sistemas de seguridad es posible que sean vulnerados y se pueda obtener información de ella y aun así la información este cifrada alguien puede tratar de descifrar esta información y obtener resultados que podrían afectar tanto a las organizaciones como a los usuarios de los servicios de estas. [5]

Otro caso notable es le sucedió a la librería electrónica de la OTAN, la cual alarmó a los suscriptores de que la información de los clientes que se encontraba en su base de datos, posiblemente había sido robada por hackers. La librería ha sido cerrada y se sugirió a los usuarios cambiar sus contraseñas si esta se usaba en otros sitios web o servicios, debido a que la información robada comprometía usuario, contraseña, dirección y correo electrónico para las personas que habían ordenado publicaciones o se habían suscrito al servicio de correo electrónico.

Lo que pone en evidencia que organizaciones importantes como lo es un servicio separado para la distribución de la información de la OTAN no está exento de tener vulnerabilidades de seguridad y por lo tanto las contraseñas que se encuentran cifradas en las diferentes bases de datos de las organizaciones pueden ser descifradas por personas malintencionadas y realizar procesos de autenticación fraudulentos que comprometan información valiosa de los diferentes usuarios del sistema.[6]

1.2 FORMULACIÓN DEL PROBLEMA

Determinación de los factores y complejidad en la decodificación de contraseñas cifradas con diferentes tipos de algoritmos a través de una validación utilizando una herramienta de decodificación de contraseñas.

1.3 DESCRIPCION DEL PROBLEMA

Muchos de los sistemas de cifrado que se han desarrollado a través de la historia han sido cuestionados. Se pone en duda la seguridad de muchos de los algoritmos debido a la corta longitud de bits de los valores generados, la facilidad para invertir el cifrado, o los posibles malos usos que se le puede dar a un algoritmo de cifrado. Estas cuestiones se suelen hacer basadas en comprobaciones matemáticas, pero también por vulneraciones que han sido llevadas a cabo en la práctica. Es por ello que surge la idea de que a través del uso de un decodificador de contraseñas se puede validar una de las propiedades que debe cumplir un algoritmo de cifrado seguro.

Se plantea el estudio para el diseño de un escenario para la validación mediante un software de decodificación de contraseñas de la seguridad de algoritmos ampliamente utilizados para la generación de valores *hash,* como un escenario que se ha visto en la vida real, donde un atacante informático tiene acceso a las contraseñas cifradas de la base de datos.

2. JUSTIFICACIÓN

Es importante conocer el nivel de seguridad que puedan tener diferentes contraseñas utilizadas por los usuarios en sistemas informáticos, comprobar la influencia de varios factores en la seguridad de contraseñas y poder dar recomendaciones basadas en experiencias prácticas para la correcta implementación y utilización de sistemas de autenticación, ya que este modo de autenticación es el más utilizado en los sistemas informáticos.

Las comprobaciones que se hacen a los algoritmos de cifrado son comprobaciones teóricas fundamentadas en conceptos matemáticos como lo es la probabilidad de colisión de hash o la complejidad computacional para hallar un valor cifrado a partir del texto plano. Estas comprobaciones son importantes pero su enfoque va dirigido únicamente al algoritmo de cifrado, dejando aparte las vulnerabilidades que puede introducir el hombre en el proceso de autenticación, pues finalmente es una persona la que realizará la asignación de contraseña y dará uso a ésta para autenticarse como usuario en algún sistema de cómputo. Es importante saber que parámetros de la asignación de contraseña tienen impacto para la decodificación del texto plano si un atacante consigue las contraseñas cifradas de un sistema de información.

Los resultados de la validación darán una orientación de la importancia de las buenas prácticas para la implementación de sistemas de autenticación mediante contraseñas para los desarrolladores de sistemas informáticos y darán una guía de buenas prácticas para los usuarios finales al momento de la creación de contraseñas. Con base en los resultados de las pruebas las organizaciones podrán darse una idea de lo que representaría el acceso de un atacante informático al texto cifrado de las contraseñas en términos de seguridad y poder realizar una correcta gestión del riesgo de seguridad orientando al usuario a correctas prácticas en la asignación de contraseñas, además de orientar la seguridad pensando más en las vulneraciones que atacan al factor humano.

Se vería reducido el impacto a los diferentes usuarios de los sistemas de información que pudiesen ser vulnerados al hacer énfasis en los factores que influyen en la decodificación de contraseñas e implementar mejores sistemas de cifrado en los procesos de autenticación.

3. OBJETIVOS

3.1 OBJETIVO GENERAL.

Validar la seguridad de las contraseñas usadas en algoritmos DES y MD5 mediante un ataque simulado a través de una herramienta de decodificación de contraseñas.

3.2 OBJETIVOS ESPECÍFICOS

- Realizar un estudio sobre los algoritmos de cifrado DES y MD5 y las características de las claves que se deben usar en su implementación.
- Realizar un estudio de las herramientas de decodificación de contraseñas.
- Diseñar un escenario de prueba para evaluar la seguridad de las contraseñas.
- Implementar el escenario y realizar un análisis de los resultados obtenidos.

4. MARCO REFERENCIAL

4.1 MARCO TEÓRICO

4.1.1 John the Ripper

John the Ripper es un rápido decodificador de contraseñas, actualmente disponible en varios sistemas operativos como derivados de Unix, Windows, DOS y OpenVMS. Su propósito principal es detectar contraseñas débiles en los sistemas Unix.

Existen dos versiones del software John the Ripper, la versión de desarrollador y la versión mejorada por la comunidad que incluye características adicionales a la versión oficial (más algoritmos de cifrado y la capacidad de poder correr el software a través de diferentes sistemas). El experimento planteado hace uso de esta característica corriendo el software en una red de computadores que es análoga a una red WAN. [7]

4.1.2 Complejidad Computacional

A los algoritmos que resuelven problemas decidibles se les puede realizar un análisis desde el punto de vista de los recursos que requiere una máquina de Turing para resolverlos. Donde nos referimos al tiempo, es decir a la cantidad de operaciones efectuadas por la máquina de Turing y el espacio, la cantidad de cinta usada por la máquina.

Este análisis sirve de comparación con otros algoritmos o para darnos una idea del tiempo que puede tardar el algoritmo en resolver algún problema según las tallas del problema.

La complejidad computacional en la criptografía es muy importante, ya que lo que se pretende es que la persona que busca la seguridad a través de este recurso pueda realizarlo de manera rápida, pero que un atacante del sistema no le sea viable o practico vulnerar la seguridad de este sistema. [8]

4.1.3 MD5

En criptografía, MD5 (abreviatura de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es un algoritmo de reducción criptográfico de

128 bits ampliamente usado. Uno de sus usos es el de comprobar que algún archivo no haya sido modificado. Generando una cadena de hash única para cada archivo que se desea comprobar. [9]

4.1.4 DES

Es un estándar para el cifrado de información, que fue desarrollado inicialmente por la compañía IBM con el nombre de LUCIFER en los años 60.Siendo uno de los primero cifradores de bloque. En los años 70 se decidió comercializar el algoritmo, donde se introdujeron algunos cambios y la NSA (Agencia de seguridad Nacional) fue uno de los más importantes contribuyentes (desde un punto de vista técnico) y no fue hasta el año 1977 que fue adoptado como un estándar por el FIPS.

Unos de los cambios que se realizaron en LUCIFER fueron muy controversiales, incluso hasta el día de hoy. Uno de los más notables es el cambio del tamaño de clave que era de 128 bits en LUCIFER, y que paso a ser de 56 bits en DES, ya que usa claves de 64 bits, pero los 8 restantes se usan para verificación de paridad. En este artículo miraremos el impacto de esta decisión y su impacto en la práctica. [10]

4.1.5 Ataques por fuerza bruta

Los ataque con fuerza bruta lo que hacen es explorar cada una de las combinaciones posibles para generar el *hash* deseado. La viabilidad de la fuerza bruta depende del dominio y el tamaño de lo contraseña [11].

En la tabla 1 podemos observar el tamaño de exploración de un ataque por fuerza bruta con diferentes posibilidades de contraseñas (ya sea que tengan solo letras minúsculas, mayúsculas y minúsculas, caracteres alfanuméricos o caracteres alfanuméricos y signos).

	min	**min/may**	**min/may/dig**	**min/may/dig/sim**
1	26	52	62	95
2	676	2704	3844	9025
4	456,976	7,311,616	14,766,336	81,450,625

8	2.09x10^11	5.35x10^13	2.18x10^14	6.63x10^15
16	4.36x10^22	2.86x10^27	4.77x10^28	4.40x10^31

Tabla 1. Espacio de búsqueda.

Un computador de escritorio puede probar un millón de contraseñas por segundo. Por lo que contraseñas muy cortas pueden ser descifradas en segundos, pero contraseñas de tamaño largo o mediano pueden tomar años sin importar que tanta capacidad computacional se tenga.

Lo que un atacante puede hacer es pre computarizar muchos valores *hash* y compararlos con los *hashes* de los usuarios para encontrar coincidencias.

4.1.6 Ataque por diccionario

Como vimos los ataques por fuerza bruta no funcionan muy bien para contraseñas de tamaño largo y mediano. Sin embargo las contraseñas actualmente son raramente generadas de manera aleatoria, en muchos de los casos, el espacio de la búsqueda es menor a 2.18x10^14, lo que significa una contraseña con 8 caracteres que pueden contener mayúsculas, minúsculas y números.

En el cable de Sony cerca del 50% de las contraseñas eran menores a 8 caracteres de longitud. Además solo el 4% de las contraseñas contenían letras mayúsculas, minúsculas y dígitos; y solo el 1% de esas contraseñas contenían caracteres no alfanuméricos.

Creando un diccionario de palabras comúnmente empleadas en contraseñas podemos acotar el espacio de búsqueda. Por ejemplo un diccionario público coincidió con el 35% de las contraseñas cableadas.

Existen muchos tipos de diccionarios. John the Ripper incluye varios que podemos utilizar, y solo basta con especificar que diccionario vamos a utilizar a través de él comando de lanzamiento del software. De esta manera no solo podremos utilizar los diccionarios provistos por Open Wall, sino que también podremos utilizar otros diccionarios, o crear uno que se ajuste más al tipo de personas o al dominio especifico que se desea vulnerar. Esto hace de la técnica algo flexible, pero al mismo tiempo incierta, ya que es posible que ninguna parte de la contraseña este contenida en el diccionario utilizado.

Por otra parte las buenas prácticas recomiendan añadir un valor salt aleatorio a la contraseña. Técnica implementada actualmente en muchos sistemas. Esto añade un nuevo reto a los atacantes, ya que tendrían que probar cada posible contraseña con cada posible salt. Y si el valor salt es lo suficientemente grande es

como si una contraseña muy débil se convirtiera en una contraseña segura de manera automática.

4.2 MARCO CONCEPTUAL

Hash: es una función que nos permite obtener un conjunto de datos L a partir de un conjunto M. En criptografía es usado para firmar documentos o realizar procesos de cifrado de contraseñas. Es importante que a partir de un elemento del conjunto M no se pueda llegar al elemento del conjunto L que genero el valor del hash. [12]

Mapeo: encontrar el valor de una función hash. [13]

Integridad: propiedad de la seguridad informática que nos asegura que la información no ha sido alterada por alguien que no debe poder alterar un archivo. [12]

Complejidad computacional: es una rama de la teoría de la computación que se centra en la clasificación de los problemas computacionales de acuerdo a su dificultad inherente, y en la relación entre dichas complejidad. Un problema se cataloga como "inherentemente difícil" si su solución requiere de una cantidad significativa de recursos computacionales, sin importar el algoritmo utilizado. La teoría de la complejidad computacional formaliza dicha aseveración, introduciendo modelos de cómputo matemáticos para el estudio de estos problemas y la cuantificación de la cantidad de recursos necesarios para resolverlos, como tiempo y memoria. [14]

Salt: En criptografía, salt es un método que cifra datos de entradas como contraseñas añadiéndoles una serie aleatoria. Esto aumenta la seguridad de las contraseñas. Hash es un método que utiliza un algoritmo para convertir datos a un valor de tamaño fijo. [15]

Crack: encontrar solución a un problema. [16]

Algoritmo: Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. [17]

Message digiest:

Las funciones de síntesis de mensajes, también llamadas funciones hash, se usan para producir resúmenes digitales de información llamados *message digests.* Los *message digests* (también llamados *hashes*) son comúnmente de 128 bits a 160 bits de longitud y proporcionan un identificador digital para cada archivo o documento digital. Las funciones de *message digest* son funciones matemáticas que procesan información para producir un resumen de mensaje diferente para

cada documento único. Los documentos idénticos tienen el mismo *message digest*; pero si incluso uno de los bits del documento cambia, el *message digest* cambia. [18]

4.3 MARCO HISTORICO

Criptoanálisis es el estudio de los métodos para obtener el sentido de una información cifrada, sin acceso a la información secreta requerida para obtener este sentido normalmente. Lo que significa vulnerar un sistema de cifrado.

Aunque el objetivo sigue siendo el mismo, los métodos y técnicas han cambiado drásticamente a través de la historia. A continuación se expondrán varios de los sucesos relevantes a través de la historia para conocer los impactos y alcances que pueden tener este tipo de ataques destinados a vulnerar sistemas de cifrado:

El criptoanálisis ha evolucionado conjuntamente con la criptografía, y la competición entre ambos puede ser rastreada a lo largo de toda la historia de la criptografía. Las claves nuevas se diseñaban para reemplazar los esquemas ya rotos, y nuevas técnicas de criptoanálisis se desarrollaban para abrir las claves mejoradas. En la práctica, se considera a ambas como las dos caras de la misma moneda: para crear un sistema criptográfico seguro, es necesario tener en cuenta los descubrimientos del criptoanálisis. De hecho, hoy en día se suele invitar a la comunidad científica a que trate de romper las nuevas claves criptográficas, antes de considerar que un sistema es lo suficientemente seguro para su uso.

La historia del criptoanálisis tiene un momento de cambio importante y sucede cuando se pasa de analizar las frecuencias de las letras y números en los mensajes de manera manual debido a lo complicado y poco efectivo que esto se volvió a medida que la complejidad de los sistemas criptográficos aumentaba. La utilización de la computación parte la historia del criptoanálisis en dos. Criptoanálisis clásico y criptoanálisis moderno, los cuales serán explicados a continuación con más detalle.

4.3.1 Criptoanálisis clásico

Aunque la expresión criptoanálisis es relativamente reciente (fue acuñada por William F. Friedman en 1920), los métodos para romper códigos y cifrados son mucho más antiguos. La primera explicación conocida del criptoanálisis se debe al sabio árabe del siglo IX, Yusuf Yaqub ibn Ishaq al-Sabbah Al-Kindi, en su Manuscrito para Descifrar Mensajes Criptográficos. Este tratado incluye una descripción del método de análisis de frecuencias.

El análisis de frecuencias es la herramienta básica para romper los cifrados clásicos. En todas las lenguas conocidas, ciertas letras del alfabeto aparecen más frecuentemente que otras; por ejemplo, en español, las vocales son muy frecuentes, ocupando alrededor del 45% del texto, siendo la E y la A las que aparecen en más ocasiones, mientras que la frecuencia sumada de F, Z, J, X, W y K no alcanza el 2%. Igualmente, se pueden reunir estadísticas de aparición de pares o tríos de letras. El análisis de frecuencias revelará el contenido original si el cifrado utilizado no es capaz de ocultar estas estadísticas. Por ejemplo, en un cifrado de substitución simple (en el que cada letra es simplemente substituida por otra), la letra más frecuente en el texto cifrado sería un candidato probable para representar la letra "E".

El análisis de frecuencias se basa tanto en el conocimiento lingüístico como en las estadísticas, pero al volverse cada vez más complicados los cifrados, las matemáticas se convirtieron gradualmente en el enfoque predominante en el criptoanálisis. Este cambio fue particularmente evidente durante la Segunda Guerra Mundial, cuando los esfuerzos para romper los códigos del Eje requirieron nuevos niveles de sofisticación matemática. Más aún, la automatización fue aplicada por primera vez en la Historia al criptoanálisis, bajo la forma de los dispositivos Bomba y Colossus, una de las primeras computadoras.

Ilustración 1. Primera página de Un manuscrito para el descifrado de mensajes criptográficos, de Al-Kindi.

4.3.2 Criptoanálisis moderno

Aunque la computación fue utilizada con gran éxito durante la Segunda Guerra Mundial, también hizo posibles nuevos métodos criptográficos que eran órdenes

de magnitud más complejos que los empleados hasta la fecha. Tomada como un todo, la criptografía moderna se ha vuelto mucho más impenetrable al criptoanalista que los métodos de pluma y papel del pasado, y parece que en la actualidad llevan ventaja sobre los métodos del puro criptoanálisis. El historiador David Kahn escribió: "Son muchos los criptosistemas en venta hoy por parte de cientos de compañías comerciales que no pueden ser rotos por ningún método conocido de criptoanálisis. De hecho, en ciertos sistemas incluso un ataque de texto plano escogido, en el que un fragmento de texto plano seleccionado es comparado con su versión cifrada, no permite conocer el código para romper otros mensajes. En cierto sentido, entonces, el criptoanálisis está muerto. Pero éste no es el final de la historia. El criptoanálisis puede estar muerto, pero, mezclando mis metáforas, hay más de un modo de desollar un gato." (Observaciones sobre el 50 Aniversario de la National Security Agency, 1 de noviembre, 2002). Kahn menciona a continuación las mayores posibilidades para la intercepción, la colocación de dispositivos grabadores ("bugging"), los ataques de canal lateral y la criptografía cuántica como sustitutos de los métodos tradicionales del criptoanálisis.

Kahn podría haberse apresurado demasiado al declarar al criptoanálisis muerto; aún no se han extinguido los cifrados débiles. En medios académicos, se presentan regularmente nuevos diseños, y también son rotos frecuentemente: el cifrado por bloques Madryga, de 1984, demostró ser vulnerable a un ataque con sólo texto cifrado disponible en 1998; FEAL-4, propuesto como sustituto para el algoritmo estándar de cifrado de datos DES fue demolido por una avalancha de ataques de la comunidad académica, muchos de los cuales no eran enteramente realizables en condiciones prácticas. En la industria, igualmente, los cifrados no están exentos de fallos: por ejemplo, los algoritmos AS/1, AS/2 y CMEA, usados en la industria de teléfonos móviles, pueden ser rotos en horas, minutos o incluso en tiempo real por equipo informático ampliamente disponible. En 2001, se demostró que el algoritmo WEP, utilizado para proteger redes Wi-Fi, es susceptible de ser atacado mediante un ataque de clave relacionada. [19]

Ilustración 2. Réplica de un dispositivo Bombe.

4.4 MARCO DE ESTADO ACTUAL

Los sistemas de cifrado están basados en el concepto de clave. La clave es la base de una transformación, normalmente matemática de un mensaje ordinario en un mensaje ilegible. La mayoría de los sistemas de cifrado están basados en un sistema de clave privada, hasta la invención de los sistemas de cifrado basados en llaves públicas en los años 70, la cual se ha convertido en una de las principales soluciones para la seguridad cibernética.

Es por ello que se realizará una comparativa que permita ver la importancia de los sistemas de cifrado asimétricos.

4.4.1 Cifrado de clave privada

Los sistemas de cifrado de clave privada utilizan una sola clave que comparten el remitente y el destinatario. Ambos deben poseer la clave; el remitente cifra el mensaje mediante la clave y el destinatario descifra el mensaje con la misma clave. Para poder establecer una comunicación privada, tanto el remitente como el destinatario deben mantener la clave en secreto. Este tipo de cifrado tiene características que lo hacen inadecuado para su uso general:

- El cifrado de clave privada requiere una clave para cada par de personas que necesitan comunicarse de forma privada. El número necesario de claves aumenta considerablemente a medida que se incrementa el número de participantes.

- Las claves se deben compartir por pares de comunicadores, por lo que las claves se deberán distribuir a los participantes. La necesidad de transmitir claves secretas las hace vulnerables al robo.
- Los participantes sólo pueden comunicarse mediante un acuerdo previo. No puede enviar un mensaje cifrado utilizable a alguien de forma espontánea. Tanto una como la otra persona deben establecer acuerdos para comunicarse compartiendo claves.

El cifrado de clave privada también se denomina cifrado simétrico, porque se utiliza la misma clave para cifrar y descifrar el mensaje.

4.4.2 Cifrado de clave pública

La seguridad de este esquema recae en la dificultad de resolución del problema de factorización de enteros y el problema del logaritmo discreto. El problema de factorización de enteros consiste en encontrar los factores primos de un numero entero positivo dado. El problema del logaritmo discreto se trata de encontrar el exponente x cuando dos elementos g y h de un grupo G son dados donde h = g^x.

Una explicación breve de como funcional seria el siguiente: el cifrado de clave pública utiliza un par de claves relacionadas matemáticamente. Un mensaje cifrado con la primera clave debe descifrarse con la segunda clave y un mensaje cifrado con la segunda clave debe descifrarse con la primera clave.

Cada participante en un sistema de claves públicas dispone de un par de claves. Una clave se designa como clave privada y se mantiene secreta. La otra clave se distribuye a quien lo desee; esta clave es la clave pública.

Cualquier usuario puede cifrar un mensaje utilizando su clave pública, pero sólo usted puede leerlo. Cuando recibe el mensaje, lo descifra utilizando la clave privada.

De forma parecida, puede cifrar un mensaje para cualquier otro utilizando su clave pública y, a continuación, descifrándola utilizando su clave privada. Entonces podrá enviar el mensaje de forma segura a través de una conexión no segura.

Este tipo de cifrado tiene características que lo hacen muy adecuado para su uso general:

- El cifrado de clave pública sólo requiere dos claves por participante.
- La necesidad de mantener el secreto es más fácil de cumplir: únicamente debe mantenerse secreta la clave privada y puesto que no necesita compartirse, es menos vulnerable al robo en la transmisión que la clave compartida en un sistema de claves simétricas.

- Las claves públicas pueden publicarse, lo que elimina la necesidad de compartir previamente una clave secreta antes de la comunicación. Cualquiera que conozca la clave pública puede utilizarla para enviar un mensaje que sólo el usuario implicado puede leer.

El cifrado de clave pública también se denomina cifrado asimétrico, porque no puede utilizarse la misma clave para cifrar y descifrar el mensaje. A cambio, se utiliza una clave de un par de claves para deshacer el trabajo del otro.

Con el cifrado de clave simétrica, debe ir con cuidado con las claves robadas o interceptadas. En el cifrado de clave pública, en el que cualquier puede crear un par de claves y publicar la clave pública, el reto consiste en verificar que identidad del propietario de la clave pública. Nada impide que un usuario cree un par de claves y publique la clave pública bajo un nombre falso. El propietario listado de la clave pública no puede leer mensajes que están cifrados con dicha clave porque el propietario no posee la clave privada correspondiente. Si el creador de la clave pública falsa puede interceptar estos mensajes, dicha persona puede descifrar y leer mensajes que están pensados para alguien más. Para contrarrestar el potencial de claves olvidadas, los sistemas de claves públicas proporcionan mecanismos para validar claves públicas e información adicional con certificados digitales y firmas digitales. [20]

4.5 MARCO CIENTIFICO

Como se expuso anteriormente uno de los más grandes avances en criptografía tienen su base en el cifrado de clave pública, la cual se basa en la dificultad de resolver el problema de la factorización de enteros y el problema del logaritmo discreto. Pero recientes investigaciones ponen en duda que este método sea del todo seguro. Ya que en el campo de la computación cuántica se han llevado a cabo importantes adelantos en los cuales la complejidad computacional de ciertos algoritmos se podría ver reducida comparada con la complejidad computacional convencional. Un claro ejemplo de ello es el algoritmo de Shor.

4.5.1 Algoritmo de Shor

En la computación convencional la pieza más pequeña de información es el bit, el cual puede tener valores de 0 de 1. En la computación cuántica esta unidad cambia a ser un qubit, el cual puede tener el valor de 0 o 1 como también ambos al mismo tiempo. Descrito como una superposición permitiría realizar millones de cálculos al mismo tiempo. [21]

En 1994 Peter Shor descubrió un algoritmo en computación cuántica que resuelve el problema de factorización de enteros y el problema del logaritmo discreto en tiempo polinomial, problemas en los que se fundamenta la criptografía asimétrica. Lo que representaría una amenaza para los sistemas de cifrado actuales, los cuales se volverían inseguros en la práctica si se llegasen a construir computadoras cuánticas lo suficientemente capaces para correr el algoritmo. Ya que se han visto importantes avances tecnológicos en el diseño de computadoras cuánticas de computadores cuánticos. Como consecuencia de ello se tendrá que dar un enfoque para solucionar los problemas que esto conlleva, lo cual se conoce como esquemas post-quantum.

4.5.2 Problemas de la computación cuántica.

Sin embargo no todo es tan bueno en la computación cuántica, ya que hay varios factores en la práctica que no se han podido solucionar en la máquina de Turing cuántica. Uno de los obstáculos principales para la computación cuántica es el problema de la decoherencia cuántica, que causa la pérdida del carácter unitario (y, más específicamente, la reversibilidad) de los pasos del algoritmo cuántico. Los tiempos de decoherencia para los sistemas candidatos, en particular el tiempo de relajación transversal (en la terminología usada en la tecnología de resonancia magnética nuclear e imaginería por resonancia magnética) está típicamente entre nanosegundos y segundos, a temperaturas bajas. Las tasas de error son típicamente proporcionales a la razón entre tiempo de operación frente a tiempo de decoherencia, de forma que cualquier operación debe ser completada en un tiempo mucho más corto que el tiempo de decoherencia. Si la tasa de error es lo bastante baja, es posible usar eficazmente la corrección de errores cuántica, con lo cual sí serían posibles tiempos de cálculo más largos que el tiempo de decoherencia y, en principio, arbitrariamente largos. Se cita con frecuencia una tasa de error límite de 10–4, por debajo de la cual se supone que sería posible la aplicación eficaz de la corrección de errores cuánticos.

Otro de los problemas principales es la escalabilidad, especialmente teniendo en cuenta el considerable incremento en qubits necesarios para cualquier cálculo que implica la corrección de errores. Para ninguno de los sistemas actualmente propuestos es trivial un diseño capaz de manejar un número lo bastante alto de qubits para resolver problemas computacionalmente interesantes hoy en día. [22]

5. DISEÑO METODOLÓGICO

5.1 MÉTODO

Inicialmente se debe documentar acerca de la seguridad informática y más específicamente en la criptografía para poder seguir el método científico, que es el más adecuado en estas comprobaciones prácticas donde uno de sus principales componentes será la medición y la comprobación empírica de conocimiento. Donde se pretende conocer la viabilidad de validar la seguridad de algoritmos de cifrado mediante el uso de un decodificador de contraseñas.

Se propone la validación de la seguridad de contraseñas a través de un decodificador de contraseñas de uso libre, tomando como muestra un conjunto de datos generados de manera pseudoaleatoria de diferentes longitudes y también contraseñas generadas a partir de palabras de común uso en el idioma inglés, midiendo los tiempos de ejecución empleados en la decodificación de cada una de las contraseñas, con una muestra de cinco elementos para cada longitud y cinco muestras para cada contraseña.

5.2 CRITERIOS DE VALIDEZ Y CONFIABILIDAD

Se realizarán pruebas para dos tipos de algoritmos (MD5 y DES), para tener puntos de comparación entre la relación de la longitud del *message digest* generado y la dificultad para romper las mismas contraseñas.

Uno de los aspectos más importantes es la realización de pruebas a cinco diferentes contraseñas de la misma longitud para descartar que debido a los procesos pseudoaleatorios en la generación de las contraseñas seguras estos pudiesen generar contraseñas que al ser decodificadas me generaran valores de tiempos muy dispersos y no se tuviera en cuenta esta variedad en el análisis de los datos.

En el diseño del escenario de validación se tuvo en cuenta que al realizar medidas siempre se genera algún tipo de error en la medición, ya que desde el momento en que se empieza a medir una variable se está afectando el sistema. Lo que se realizó para disminuir el error en las medidas fue tomar cinco tiempos diferentes para la misma contraseña, lo cual se demostrara posteriormente que los valores de error absoluto quedaron en su mayoría con valores entre 0.001 y 0.3 segundos. Los errores relativos oscilaran entre el 0.0015 y el 3% a excepción de varias medidas atípicas.

Se validan diferentes tipos de contraseñas, las contraseñas seguras generadas de manera pseudoaleatoria y contraseñas generadas a partir de las palabras más comunes del idioma inglés.

5.3 DEFINICION DE HIPÓTESIS

Es posible realizar una validación de la seguridad de contraseñas que han sido cifradas con diferentes algoritmos mediante la utilización de un decodificador de contraseñas.

5.4 VARIABLES E INDICADORES

Las principales variables en del diseño del escenario de validación son: el tiempo que tarda la aplicación de software utilizado en la decodificación de una contraseña específica, la longitud de las diferentes contraseñas (de 1 a 5 caracteres), variedad en las contraseñas de una misma longitud, el uso de dos algoritmos de cifrado con tamaños de *message digest* diferentes, el uso de diferentes capacidades de computo en la realización de las decodificaciones.

5.5 UNIVERSO O POBLACIÓN

El universo son los posibles *message digest* generados a partir de diferentes algoritmos de cifrado.

5.6 MUESTRA

Se tomó una muestra de 50 contraseñas diferentes según los criterios de validez explicados anteriormente para ser cifrados con los algoritmos MD5 y DES generando sus respectivos valores de *message digest*.

6. DESARROLLO

6.1 Red

Se realiza el montaje de la red que se observa en la Ilustración 3 donde se correrán y realizaran diferentes pruebas con el software JtR.

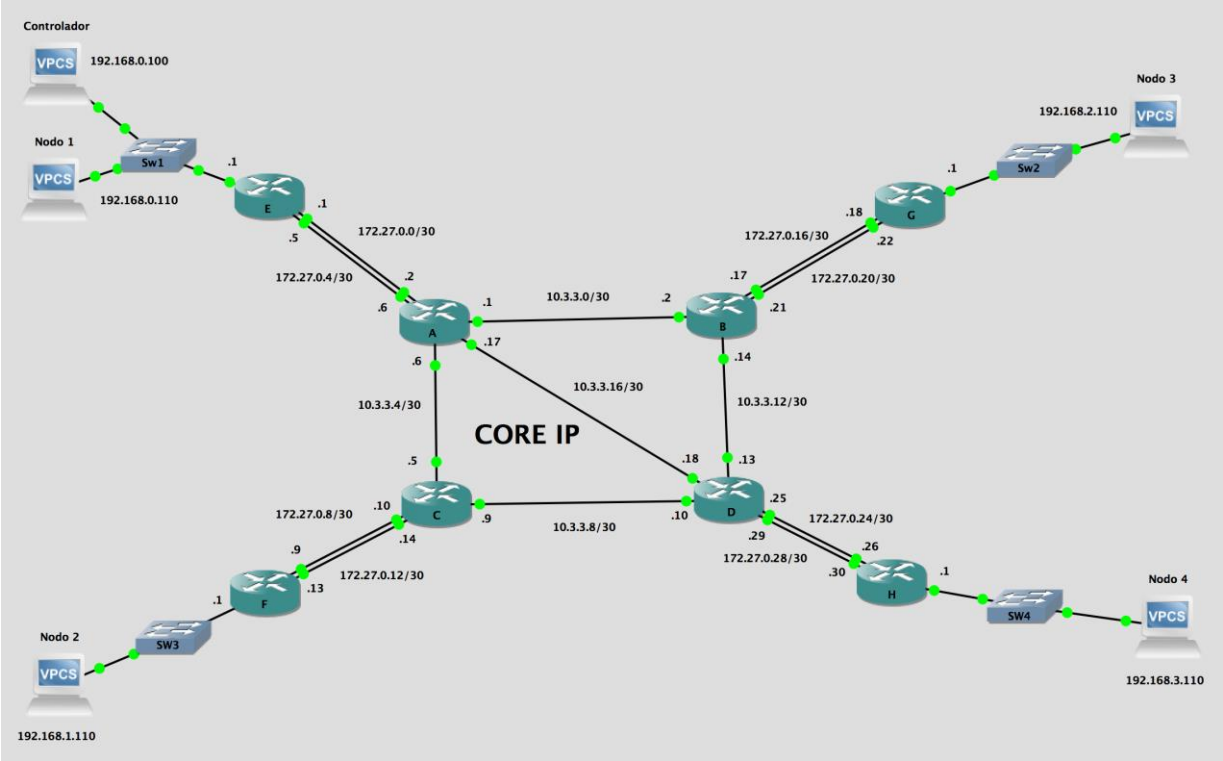

Ilustración 3. Topografía de la red.

En esta red el nodo maestro será el computador con la ip 192.168.0.100, y los demás nodos serán nodos esclavos. Donde se utilizaron los switch: Cisco SG220-26 26-Port Gigabit Smart Plus Switch.

6.2 Nodos

Cada uno de los nodos de la red se trabajó como máquinas virtuales, con el uso del software VirtualBox, dándoles a cada una de estas una memoria RAM de 2 Gb, y el uso de los 4 núcleos de la computadora.

En cada una de estas máquinas virtuales se instaló el sistema operativo Ubuntu 16.04 LTS, la última versión de g++ y gcc, el cliente y servidor ssh, Portmap y NFS para acceder a ficheros remotos como si fueran locales, Openssl y diferentes componentes SSL necesarios para hacer uso de MPI un software que nos ayudará a correr JtR a través de cada una de las máquinas de manera tan transparente como si se corriera en una sola máquina y por último se realizó la instalación de la versión mejorada por la comunidad de JtR (john-1.8.0-jumbo-1).

El nodo Maestro cobra gran importancia, ya que desde este se debe poder realizar una conexión ssh sin solicitar una contraseña, y cada uno de los nodos esclavos realizara un montaje de un directorio de archivos perteneciente a este nodo principal.

6.2.1 Características de los equipos

Los computadores utilizados fueron cinco Dell Optiplex 3020, Intel Core i5-4590 CPU @ 3.30GHz 3.30GHz, con 8 gigas de Ram y con el sistema operativo Windows 7 Professional.

6.2.2 Características de la máquina virtual

Todos los nodos corrían la misma máquina virtual, por lo tanto tenían las mismas características, las cuales se pueden observar en la Ilustración 4 donde se presenta la configuración de una de ellas en el Oracle VM.

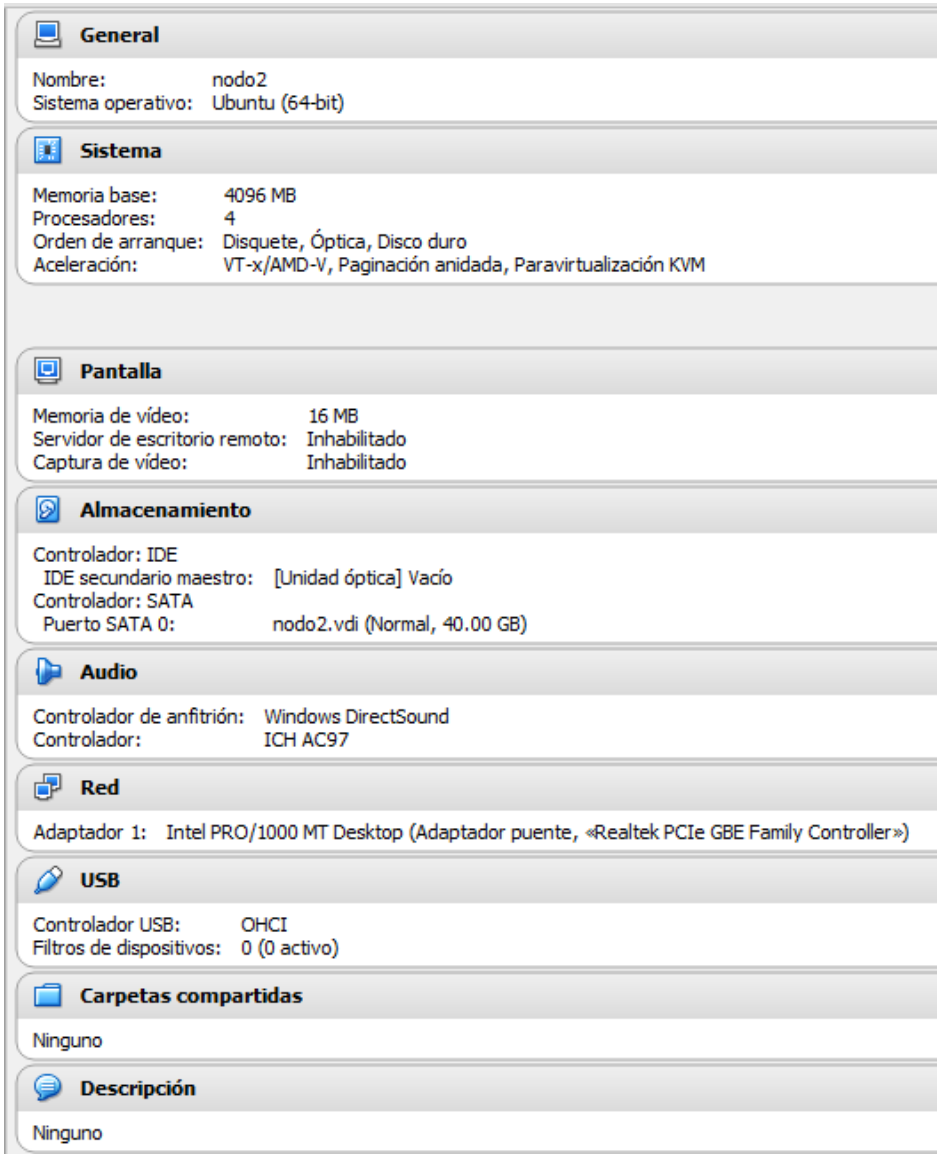

Ilustración 4. Características de los equipos.

6.3 Generación de contraseñas

Se generaron contraseñas de diferentes longitudes, empezando en 1 y terminando en 5. Donde se generaron dos tipos de contraseñas.

6.3.1 Contraseñas seguras

Para la generación de contraseñas seguras se utilizó el software PWGen el cual crea contraseñas seguras de manera pseudoaleatorea, ya que los humanos no somos muy buenos generando contraseñas que cumplan algunos de los criterios más sencillos de aleatoriedad. Es por eso que esta herramienta nos brinda la posibilidad de hacerlo. Esta herramienta genera contraseñas con minúsculas, mayúsculas y números.

Se plantea la dificultad de recordar estas contraseñas largas y aleatorias para las personas, por ello también nos brinda la posibilidad de crear *passphrase*, que son contraseñas creadas a partir de diferentes palabras y solo cambiando unas cuantas letras y agregando números. Se sostiene la idea de que de que la mayoría de las personas es capaz de memorizar una contraseña de 90 bits (aproximadamente 11 caracteres) con algo de esfuerzo y que no es tan difícil memorizar una contraseña aleatoria. [23]

6.3.2 Contraseñas débiles

Para simular contraseñas débiles se utilizaron las palabras más comunes del idioma inglés. [24]

En la Tabla 2 se presenta un listado de las 100 palabras más comunes del idioma inglés

1	the	26	they	51	when	76	come
2	be	27	we	52	make	77	its
3	to	28	say	53	can	78	over
4	of	29	her	54	like	79	think
5	and	30	she	55	time	80	also
6	a	31	or	56	no	81	back
7	in	32	an	57	just	82	after
8	that	33	will	58	him	83	use
9	have	34	my	59	know	84	two
10	I	35	one	60	take	85	how
11	it	36	all	61	people	86	our
12	for	37	would	62	into	87	work
13	not	38	there	63	year	88	first
14	on	39	their	64	your	89	well
15	with	40	what	65	good	90	way
16	he	41	so	66	some	91	even
17	as	42	up	67	could	92	new
18	you	43	out	68	them	93	want
19	do	44	if	69	see	94	because
20	at	45	about	70	other	95	any
21	this	46	who	71	than	96	these
22	but	47	get	72	then	97	give
23	his	48	which	73	now	98	day
24	by	49	go	74	look	99	most
25	from	50	me	75	only	100	us

Tabla 2. Palabras más comunes del inglés.

6.4 Generación de valores *Hash*

Los valores de *hash* a partir de las contraseñas generadas se generaron con la herramienta Openssl. Con el comando passwd se generaron diversos *hash* con diferentes algoritmos de cifrado.

Posteriormente se guardaban en un archivo de texto con el nombre de la contraseña ingresada.

6.4.1 Generación de valores *hash* MD5

Se generaron valores *hash* con el algoritmo MD5 sin salt, utilizando el siguiente comando:

Openssl passwd -1 password

6.4.2 Generación de valores *hash* DES

Se generaron valores *hash* con el algoritmo crypt (una función para la generación de *hash*es en Unix basada en DES) sin salt, utilizando el siguiente comando:

Openssl passwd password

6.5 Realización de pruebas

Para la realización de pruebas se lanzó el software JtR de manera distribuida a cada uno de los nodos de la red planteada anteriormente desde el nodo maestro (o nodo controlador), haciendo uso de MPI y NFS. Esto se realizó con el siguiente comando:

Time mpiexec –hostfile mpi-nodes.txt ./john /var/mpishare/hash –pot=/var/mpishare/sharedpot

Donde en *mpi-nodes.txt* encontramos las direcciones ip de cada uno de los nodos de la red y la cantidad de CPU a usar en cada nodo, *hash* es alguno de los *hashes* generados que se quería descifrar y *sharedpot* es un archivo que contiene las contraseñas descifradas, el cual siempre estaba vacío en cada una de las pruebas.

Es importante resaltar que estas pruebas se hicieron haciendo uso del diccionario que usa por defecto JtR, y que algunas pruebas por sencillas que parezcan no son tan triviales de resolver utilizando este método de ataques por diccionario, ya que hay algunas soluciones que podrían ser mucho más rápidas si se resuelven totalmente a fuerza bruta. JtR nos permite realizar un ataque totalmente a fuerza bruta con el modo incremental, pero nos quisimos hacer uso de él para que las pruebas fueran más uniformes y se pudieran comprar, además para realizar una medición de la efectividad del uso de diccionarios en contraseñas generadas de manera pseudoaleatoria.

6.6 Medición de tiempos

Cada uno de los tiempos fue tomado con la herramienta time de que nos proporciona la terminal de comandos de Linux. Este comando nos arroja 3 valores, pero el tenido en cuenta para todas las mediciones fue el real, ya que este toma en cuenta el tiempo gastado en la realización de cada una de las conexiones y la escritura de los datos.

7. RESULTADOS

7.1 Utilizando dos CPU

En la Tabla 3,Tabla 4 y Tabla 5 se presentan los tiempos medidos para el descifrado de contraseñas seguras cifradas con MD5, para contraseñas seguras cifradas con DES, para contraseñas débiles cifradas con MD5 respectivamente, cada una de ellas haciendo uso de dos CPU de cada máquina.

	MD5-BSD con dos CPU					
	Pass	1	2	3	4	5
1	q	1m4.172s	0m47.522s	0m57.425s	0m58.096s	0m59.297s
	v	0m54.818s	1m7.432s	1m6.941s	1m10.968s	1m15.449s
	p	0m19.435s	0m20.398s	0m21.232s	0m18.578s	0m18.907s
	x	0m36.865s	0m35.453s	0m35.724s	0m35.800s	0m35.584s
	f	0m36.661s	0m35.172s	0m36.557s	0m34.823s	0m34.823s
2	ee	1m3.728s	1m3.829s	1m3.921s	1m4.049s	1m3.767s
	HL	35m3.763s	35m3.598s	35m3.363s	35m3.798s	35m3.732s
	52	4m48.975s	4m49.516s	5m3.857s	4m48.706s	4m49.573s
	Cc	8m3.940s	8m4.218	8m25.223s	8m7.596s	8m24.275
	pP	>500m				
3	Bei	40m3.834s				
	2li	>500m				
	eYB	>500m				
	q80	>500m				
	ugo	>500m				
4	ii0m	>1181m35.533s				
	MDm2	>500m				
	s7gO	>500m				
	M5wA	>500m				
	9riA	>500m				

Tabla 3. Mediciones de contraseñas seguras MD5 dos CPU.

	DES con 2 CPU					
	pass	1	2	3	4	5
1	q	0m5.438s	0m5.236s	0m5.415s	0m5.748s	0m5.531s
	v	0m4.370s	0m5.372s	0m5.306s	0m5.462s	0m5.666s
	p	0m4.170s	0m3.809s	0m3.793s	0m4.131s	0m3.823s
	x	0m4.357s	0m4.108s	0m4.048s	0m3.442s	0m3.786s
	f	0m4.404s	0m3.734s	0m3.967s	0m4.578s	0m3.880s
2	ee	0m4.387s	0m5.635s	0m5.012s	0m5.253s	0m5.523s
	HL	0m32.474s	0m33.979s	0m32.132s	0m33.442s	0m33.331s
	52	0m15.062s	0m15.898s	0m15.700s	0m14.730s	0m14.591s
	Cc	0m18.865s	0m17.928s	0m17.950s	0m17.441s	0m16.448s
	pP	>200m	>200m	>200m	>200m	>200m
3	Bei	0m35.854s	0m35.482s	0m35.934s	0m35.343s	0m35.438s
	2li	1m0.389s	1m0.434s	1m0.394s	1m0.3239s	1m0.840s
	eYB	6m26.814s	6m26.493s	6m26.438s	6m26.439s	6m26.948s
	q80	1m22.726s	1m22.239s	1m22.923s	1m22.329s	1m22.239s
	ugo	0m49.385s	0m49.329s	0m49.329s	0m49.940s	0m49.379s
4	ii0m	0m52.918s	0m52.918s	0m52.918s	0m52.918s	0m52.918s
	MDm2	3m20.929s	3m20.329s	3m20.328s	3m20.943s	3m20.439s
	s7gO	12m43.793s	12m43.329s	12m43.329s	12m43.239s	12m43.439s
	M5wA	9m14.744s	9m14.934s	9m14.380s	9m14.832s	9m14.298s
	9riA	9m33.399s	9m33.399s	9m33.399s	9m33.399s	9m33.399s

Tabla 4. Mediciones de contraseñas seguras DES dos CPU.

	MD5-GEN common words 2CPUS					
	pass	1	2	3	4	5
1	a	0m4.554s	0m3.852s	0m3.915s	0m3.710s	0m3,284s
	I	1m3.864s	1m3.848s	1m3.866s	1m3.685s	1m4.122s
2	to	0m16.358s	0m17.727s	0m16.519s	0m18.964s	0m18.033s
	of	2m3.788s	2m3.781s	2m3.725s	2m3.812s	2m3.834s
	in	16m3.941s	16m3.794s	16m3.235s	16m3.832s	16m3.394s
	it	16m3.830s	16m3.925s	16m3.943s	16m3.934s	16m3.823s
	on	11m3.870s	11m3.837s	11m3.348s	11m3.945s	11m3.834s
3	the	0m4.578s	0m3.996s	0m3.715s	0m3.404s	0m3.514s
	and	0m9.926s	0m9.808s	0m9.405s	0m10.070s	0m9.820s
	for	3m3.656s	3m3.802s	3m3.703s	3m3.633s	3m3.987s
	not	2m3.717s	2m3.781s	2m3.798s	2m4.856s	2m3.794s
	you	3m36.847s	3m35.238s	3m36.651s	3m35.361s	3m35.625s
4	that	0m48.763s	0m49.654s	0m48.491s	0m48.555s	0m48.059s
	with	30m3.643s	30m3.934s	30m3.348s	30m3.348s	30m3.347s
	this	0m48.776s	0m49.159s	0m48.966s	0m48.234s	0m49.722s
	from	6m41.001s	6m38.514s	6m39.622s	6m39.702s	6m39.541s
	they	0m47.601s	0m49.240s	0m48.560s	0m48.978s	0m49.677s
5	would	33m17.522s	33m17.348s	33m17.348s	33m17.834s	33m17.348s
	there	0m27.097s	0m26.285s	0m24.764s	0m25.391s	0m25.445s
	their	5m22.153s	5m28.946s	5m3.977s	5m5.518s	5m6.639s
	about	0m26.512s	0m27.642s	0m26.472s	0m27.052s	0m26.290s
	which	34m3.644s	34m3.348s	34m3.348s	34m3.348s	34m3.349s

Tabla 5. Mediciones contraseñas débiles MD5 dos CPU.

7.2 Utilizando cuatro CPU

En la Tabla 6 y Tabla 7 se presentan las mediciones con 4 CPU.

	DES con 4 CPU					
	pass	1	2	3	4	5
1	q	0m2.860s	0m2.796s	0m3.087s	0m3.029s	0m2.884s
	v	0m2.917s	0m2.858s	0m2.855s	0m2.793s	0m2.975s
	p	0m1.855s	0m1.171s	0m1.845s	0m1.814s	0m1.710s
	x	0m2.260s	0m2.148s	0m2.158s	0m2.280s	0m2.187s
	f	0m2.150s	0m2.283s	0m2.291s	0m2.081s	0m2.220s
2	ee	0m2.817s	0m2.922s	0m2.970s	0m2.817s	0m2.959s
	HL	0m13.160s	0m11.700s	012.143s	0m12.419s	0m12.209s
	52	0m4.834s	0m4.726s	0m4.645s	0m4.601s	0m4.464s
	Cc	0m5.706s	0m5.805s	0m5.726s	0m5.827s	0m5.989s
	pP	183m0.490s	183m0.479s	182m0.497s	185m0.509s	184m1.208s
3	Bei	0m14.004s	0m14.007s	0m13.152s	0m14.239s	0m14.238s
	2li	0m22.761s	0m22.837s	0m23.378s	0m22.238s	0m22.349s
	eYB	2m24.344s	2m25.376s	2m24.052s	2m24.238s	2m24.329s
	q80	0m31.338s	0m30.905s	0m31.384s	0m31.239s	0m31.349s
	ugo	0m19.000s	0m18.950s	0m19.040s	0m19.934s	0m19.349s
4	ii0m	0m19.849s	0m20.199s	0m20.003s	0m20.234s	0m20.239s
	MDm2	1m16.094s	1m14.939s	1m16.382s	1m16.239s	1m16.239s
	s7gO	4m46.326s	4m41.619s	4m39.779s	4m40.349s	4m40.238s
	M5wA	3m30.349s	3m28.661s	3m28.772s	3m28.239s	3m28.239s
	9riA	3m25.446s	3m29.005	3m34.502s	3m32.449s	3m30.235
5	yee7A	2m55.339s	2m54.846s	2m54.858s	2m55.738s	2m54.466s
	Eij2u	78m0.440s	77m0.441s	79m0.615s	77m0.420s	78m0.535s
	eeJ6E	1054m1.234s	1041m1.297s	1039m1.199s	1041m1.241s	1043m15.075s
	Buef6	1m34.496s	1m36.882s	1m34.164s	1m33.867s	1m35.321s
	lad9Y	230m0.518s	229m0.547s	228m0.563s	228m0.588s	228m0.540s

Tabla 6. Mediciones de contraseñas seguras DES con 4 CPU.

	DES common words 4 CPUS					
	pass	1	2	3	4	5
1	a	0m0.561s	0m0.489s	0m0.342s	0m0.320s	0m0.353s
	l	0m2.984s	0m2.934s	0m2.921s	0m2.920s	0m2.772s
2	to	0m1.586s	0m1.783s	0m1.683s	0m1.675s	0m1.747s
	of	0m3.740s	0m3.575s	0m3.665s	0m3.747s	0m3.656s
	in	0m8.573s	0m8.330s	0m8.561s	0m8.754s	0m8.404s
	it	0m8.325s	0m8.290s	0m9.116s	0m8.524s	0m8.790s
	on	0m7.661s	0m7.026s	0m6.905s	0m7.220s	0m7.198s
3	the	0m0.333s	0m0.351s	0m0.338s	0m0.323s	0m0.326s
	and	0m1.917s	0m1.553s	0m1.605s	0m1.634s	0m1.547s
	for	0m3.765s	0m4.212s	0m3.719s	0m3.799s	0m3.756s
	not	0m3.272s	0m3.455s	0m3.451s	0m3.228s	0m3.209s
	you	0m4.096s	0m4.169s	0m4.102s	0m4.020s	0m3.989s
4	that	0m2.534s	0m2.484s	0m2.464s	0m2.506s	0m2.473s
	with	0m11.083s	0m10.988s	0m10.945s	0m10.871s	0m11.228s
	this	0m2.622s	0m2.542s	0m2.627s	0m2.507s	0m2.676s
	from	0m5.127s	0m5.157s	0m5.040s	0m5.034s	0m5.112s
	they	0m2.580s	0m2.612s	0m2.606s	0m2.610s	0m2.589s
5	would	0m12.396s	0m11.688s	0m12.188s	0m11.760s	0m11.692s
	there	0m2.147s	0m2.016s	0m2.099s	0m2.080s	0m2.007s
	their	0m4.705s	0m4.639s	0m5.065s	0m4.684s	0m4.661s
	about	0m2.020s	0m2.004s	0m2.052s	0m2.005s	0m1.959s
	which	0m11.749s	0m11.939s	0m11.710s	0m12.281s	0m11.989s

Tabla 7. Mediciones de contraseñas débiles DES con 4 CPU.

7.3 Análisis de los datos

Es importante notar que hay contraseñas con tiempos irregulares, que toman más tiempo que contraseñas de la misma longitud y cifradas de igual manera.

Como también algunos datos que tuvieron que ser omitidos debido a lo poco práctico que era realizar las mediciones de tiempos muy elevados, donde en muchos casos lo más adecuado fue finalizar una ejecución sin hallar la contraseña que generaba el respectivo *hash*.

Para un mejor tratamiento de los datos y un fácil análisis se trabajan las pruebas en segundos, tomando como la mejor estimación del valor verdadero el valor promedio de las medidas para cada una de las contraseñas, el cual fue calculado con la Ecuación 1.

$$< x >= \frac{x_1 + x_2 + \ldots x_n}{n} = \frac{\sum_1^n x_i}{n}$$

Ecuación 1. Valores verdaderos.

Calculamos el error absoluto utilizando la Ecuación 2.

$$\Delta x = \sqrt{\frac{\sum_1^n (x_i - < x >)^2}{n(n-1)}}$$

Ecuación 2. Error absoluto.

El error relativo fue calculado con la Ecuación 3.

$$e = \frac{\Delta x}{< x >}$$

Ecuación 3. Error relativo.

En la Tabla 8, Tabla 9, Tabla 10 y Tabla 11 se presentan los valores verdaderos, errores absolutos y relativos calculados como se describió anteriormente.

Valores contraseñas seguras, DES con dos CPU

longitud	pass	media	abs	relativo
1	q	5.474	0.084	1.527%
	v	5.235	0.225	4.291%
	p	3.945	0.084	2.134%
	x	3.948	0.156	3.945%
	f	4.113	0.161	3.921%
2	ee	5.162	0.222	4.300%
	HL	33.072	0.337	1.018%
	52	15.196	0.260	1.708%
	Cc	17.726	0.394	2.222%
	pP	---	---	---
3	Bei	35.610	0.119	0.333%
	2Ii	60.476	0.093	0.153%
	eYB	386.626	0.107	0.028%
	q80	82.491	0.141	0.170%
	ugo	49.472	0.118	0.238%
4	ii0m	52.918	0.001	0.002%
	MDm2	200.594	0.141	0.070%
	s7gO	763.426	0.097	0.013%
	M5wA	554.638	0.126	0.023%
	9riA	573.399	0.001	0.000%

Tabla 8. Valores verdaderos contraseñas seguras DES con dos CPU.

Valores palabras comunes, MD5 con dos CPU.

longitud	pass	media	abs	relativo
1	a	3.863	0.205	5.302%
	I	63.877	0.070	0.110%
2	to	17.520	0.487	2.780%
	of	123.788	0.018	0.015%
	in	963.639	0.137	0.014%
	it	963.891	0.027	0.003%
	on	663.767	0.107	0.016%
3	the	3.841	0.210	5.464%
	and	9.806	0.111	1.129%
	for	183.756	0.065	0.035%
	not	123.989	0.217	0.175%
	you	215.944	0.336	0.156%
4	that	48.704	0.264	0.541%
	with	1803.524	0.117	0.007%
	this	48.971	0.243	0.496%
	from	399.676	0.395	0.099%
	they	48.811	0.353	0.723%
5	would	1997.480	0.095	0.005%

	there	25.796	0.405	1.571%
	their	313.447	5.074	1.619%
	about	26.794	0.247	0.923%
	which	2043.407	0.059	0.003%

Tabla 9. Valores verdaderos palabras comunes MD5 con dos CPU.

Valores contraseñas seguras, DES con cuatro CPU

longitud	pass	media	abs	relativo
1	q	2.931	0.055	1.86%
	v	2.880	0.031	1.07%
	p	1.679	0.130	7.72%
	x	2.207	0.027	1.22%
	f	2.205	0.040	1.82%
2	ee	2.897	0.034	1.16%
	HL	12.326	0.239	1.94%
	52	4.654	0.062	1.33%
	Cc	5.811	0.050	0.86%
	pP	11004.637	30.639	0.28%
3	Bei	13.928	0.201	1.44%
	2Ii	22.713	0.202	0.89%
	eYB	144.468	0.233	0.16%
	q80	31.243	0.088	0.28%
	ugo	19.255	0.184	0.95%
4	ii0m	20.105	0.077	0.38%
	MDm2	75.979	0.264	0.35%
	s7gO	281.662	1.205	0.43%

	M5wA	208.852	0.390	0.19%
	9riA	210.325	1.542	0.73%
5	yee7A	175.049	0.221	0.13%
	Eij2u	4668.490	22.483	0.48%
	eeJ6E	62620.009	160.417	0.26%
	Buef6	94.946	0.542	0.57%
	Iad9Y	13716.551	23.991	0.17%

Tabla 10. Valores verdaderos contraseñas seguras DES con cuatro CPU.

Valores palabras comunes, DES con cuatro CPU

longitud	pass	media	abs	relativo
1	a	0.413	0.047	11.481%
	l	2.906	0.036	1.222%
2	to	1.695	0.034	1.995%
	of	3.677	0.032	0.857%
	in	8.524	0.074	0.865%
	it	8.609	0.155	1.798%
	on	7.202	0.128	1.784%
3	the	0.334	0.005	1.482%
	and	1.651	0.068	4.143%
	for	3.850	0.091	2.372%

	not	3.323	0.054	1.627%
	you	4.075	0.032	0.784%
4	that	2.492	0.013	0.505%
	with	11.023	0.062	0.559%
	this	2.595	0.031	1.184%
	from	5.094	0.024	0.479%
	they	2.599	0.006	0.243%
5	would	11.945	0.146	1.223%
	there	2.070	0.026	1.267%
	their	4.751	0.079	1.670%
	about	2.008	0.015	0.748%
	which	11.934	0.102	0.854%

Tabla 11. Valores verdaderos contraseñas débiles DES con cuatro CPU.

7.4 Contraseñas seguras, MD5 con dos CPU

En estos datos podemos observar que el software JtR puede romper contraseñas muy cortas en menos de un segundo pero a medida que el tamaño de las contraseñas va creciendo, se hace poco práctico la decodificación de estas, es por eso que muchos de los experimentos no tuvieron un final. Es importante notar que contraseñas de 3 caracteres sigue siendo una contraseña muy pequeña, que a fuerza bruta (con el modo incremental de JtR) esta tarea se hubiese podido realizar en menos de un minuto.

Sin embargo, podemos evidenciar que el tiempo de solución crece de manera muy apresurada a medida que la longitud de la contraseña va creciendo, esta relación de crecimiento la podremos analizar en los siguientes experimentos, donde fue más viables finalizar muchas de las pruebas propuestas, utilizando una capacidad de computo mayor.

7.5 Contraseñas seguras, DES con dos CPU

A diferencia de las pruebas para el algoritmo MD5, pudimos terminar todas las pruebas para contraseñas de menos de 5 caracteres. Esto debido a que los

hashes generados por DES tienen una menor longitud y no hace uso de tantas funciones Feistsel. En la búsqueda a fuerza bruta para DES el dominio de exploración es de 2^64, mientras que el tamaño de exploración para el MD5 es de 2^128.

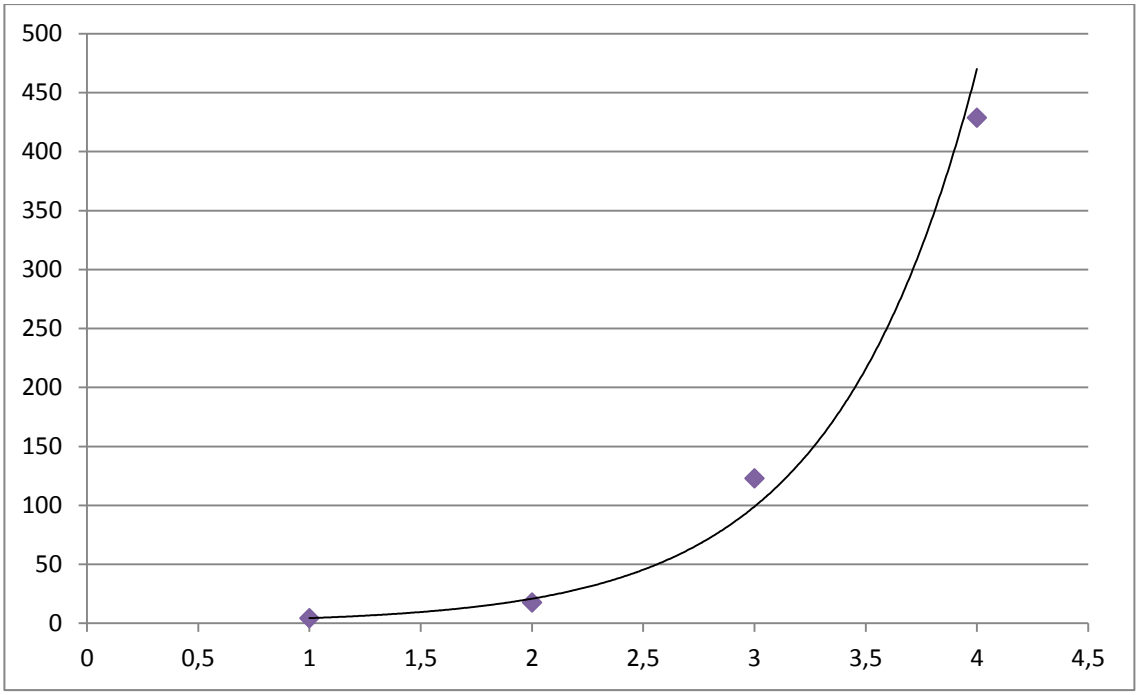

Ilustración 5. Tiempos para contraseñas seguras DES dos CPU.

En la Ilustración 5 vemos como crece el tiempo (eje y) necesario para hallar las diferentes contraseñas según su longitud (eje x). Corresponde a la gráfica de una función exponencial. Una gráfica muy similar a esta sería la gráfica de un ataque a fuerza bruta.

Con esto se puede observar que los ataques de diccionario tienen la misma tendencia de crecimiento (O (2^n), donde en es la cantidades de bits de la contraseña) que un ataque a fuerza bruta si las contraseñas generadas son aleatorias.

7.6 Palabras comunes, MD5 con dos CPU

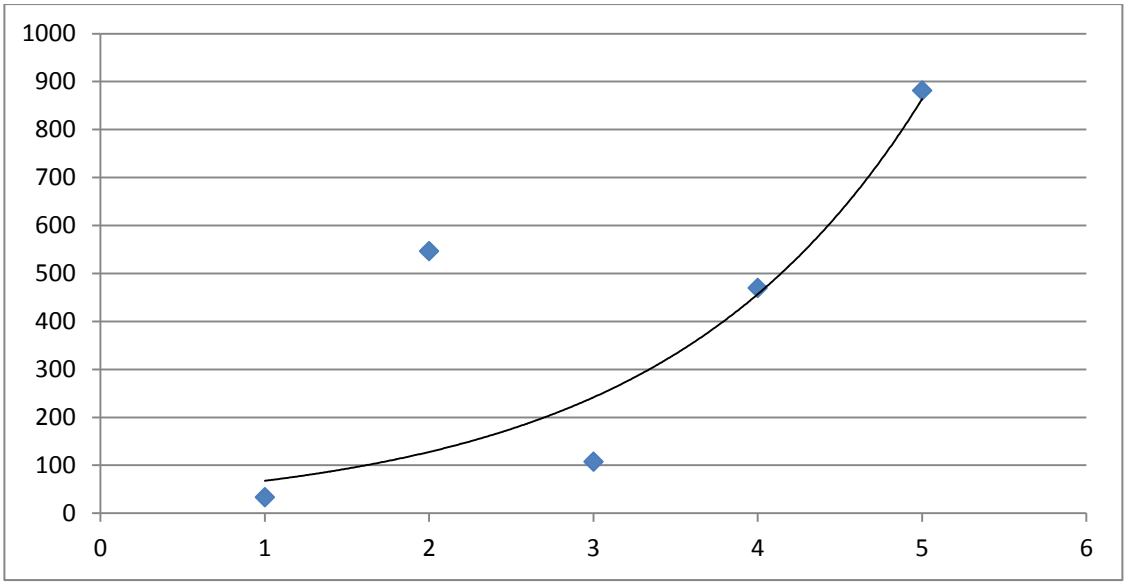

Ilustración 6. Tiempos para contraseñas débiles MD5 dos CPU.

En la Ilustración 6 podemos ver que la tendencia de crecimiento del tiempo según la longitud de la contraseña no es perfectamente exponencial, aunque pueda parecer que exista una tendencia a ello.

Que la distribución no siga una tendencia exacta se puede deber a que las pruebas se realizan según el orden del diccionario, donde las palabras con más probabilidad son probadas son probadas inicialmente. Esto hace que la longitud de la contraseña influya debido a que es pertinente probar contraseñas cortas inicialmente para descartarlas, pero es muy probable que palabras que pueden ser largas tengan mayor precedencia que palabras que son más cortas pero de menor uso.

7.7 Contraseñas seguras, DES con cuatro CPU

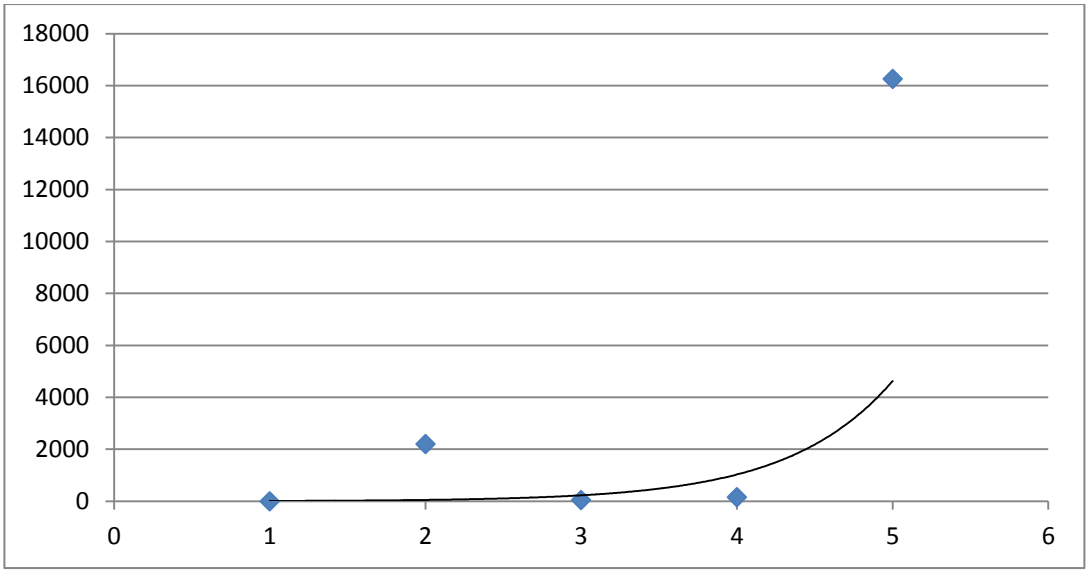

Ilustración 7. Tiempos para contraseñas seguras DES con CPU.

Al observar la Ilustración 7 se evidencia que al utilizar 4 CPU para el mismo algoritmo de cifrado se disminuyó bastante los tiempos de ejecución para la decodificación de contraseñas seguras, permitiendo que fuera practico realizar pruebas para contraseñas de cinco caracteres.

El tiempo promedio tardado para descifrar contraseñas de cinco caracteres es muy grande y se sale de la tendencia exponencial. También vemos un pequeño brinco para contraseñas de dos caracteres, esto es debido a que la contraseña pP nos genera un dato atípico que sube el promedio de los tiempos. Pero en términos generales se presenta la misma tendencia que vimos con los tiempos de descifrado de DES con dos CPU.

7.8 Palabras comunes, DES con cuatro CPU

Los resultados de estas pruebas son unos de los más interesantes, donde se evidencia que para contraseñas cortas cifradas con DES podemos hallar la solución de manera muy eficiente. Reafirmamos algunas de las críticas que ha tenido el algoritmo en cuanto al tamaño del *hash* generado y comprobamos la real eficiencia del software John the Ripper contra el estándar de cifrado de contraseñas de Unix. También se atribuye la eficiencia de la decodificación de contraseñas a la debilidad de estas.

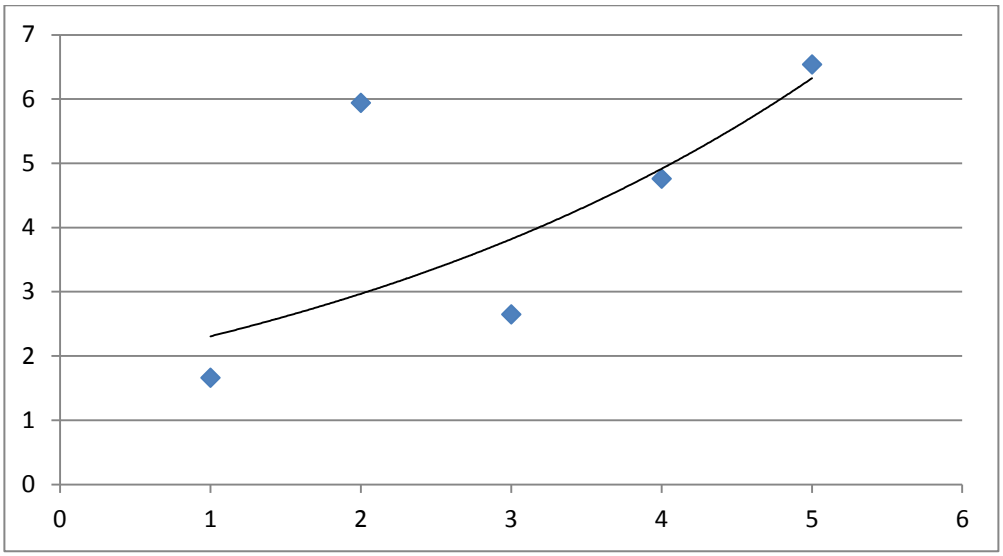

Ilustración 8. Tiempos para contraseñas débiles DES con cuatro CPU.

8. CONCLUSIONES

- Se recomienda no usar DES para generar *hashes* de contraseñas, se pudo observar de forma práctica que estos son muy cortos.
- Hay una tendencia exponencial en el aumento del tiempo para descifrar contraseñas respecto a la longitud de estas, algo similar a la complejidad computacional teórica de los ataques por fuerza bruta que es O (2^n).
- Es fundamental hacer uso de los *salt*, ya que estos aseguran que el *hash* generado va a provenir de una contraseña segura, ya que si el usuario ingresa una contraseña débil al agregarle un valor aleatorio esta nueva contraseña será una contraseña segura.
- Se recomienda que las contraseñas tengan un tamaño mínimo de 8 caracteres en donde se incluyan minúsculas, mayúsculas y numero, ya que a partir de este espacio de búsqueda se hace poco práctico la decodificación de la contraseña, siendo el espacio de búsqueda muy grande (2.18x10^14).
- Estos sistemas serán inseguros contra criptoanálisis post-cuántico.

9. RECOMENDACIONES.

Se recomienda desarrollar un escenario de validación similar al expuesto en el trabajo de grado donde se realice inicialmente ataques por fuerza bruta de manera incremental hasta cierta longitud de contraseñas y después realizar un ataque por diccionario, así como también probar otros algoritmos más seguros como el SHA-2.

Se propone realizar cálculos del tiempo estimado de vulneración de sistemas de autenticación suponiendo que se solucionan los problemas que presenta actualmente la computación cuántica, como un estudio del impacto que tendrían los sistemas de información cuando esto suceda.

Adaptar el escenario de validación para la realización de pruebas en varios sistemas de información que deseen probar la seguridad de las contraseñas utilizadas por sus usuarios, todo con el fin de ver las vulnerabilidades que tiene las empresas en la práctica. Para ello sería recomendado el desarrollo de un diccionario específico para las condiciones de los empleados y la empresa, donde sería importante tener en cuenta fechas de cumpleaños y nombres de los usuarios.

10. DIVULGACIÓN

Se participó en el Simposio de Investigación Ustamed 2017 con la ponencia intitulada “Una mirada a la complejidad computacional y seguridad en la práctica de los algoritmos MD5 y DES.”, realizado en la ciudad de Medellín el día 10 de noviembre de 2017 en la Universidad Santo Tomás sede Medellín.

11. BIBLIOGRAFIA

[1]"Statistics", *ITU*, 2017. [Online]. Available: http://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx. [Accessed: 28- Nov- 2017].

[2]J. Campos and García, "Gráficos sobre la brecha digital en el mundo en 2015", *La Vanguardia*, 2017. [Online]. Available: http://www.lavanguardia.com/vangdata/20150529/54431507120/graficos-brecha-digital-en-mundo-2015.html. [Accessed: 28- Nov- 2017].

[3]"La importancia de la Seguridad Informática", *Trustdimension.com*, 2017. [Online]. Available: http://www.trustdimension.com/la-importancia-de-la-seguridad-informatica/. [Accessed: 28- Nov- 2017].

[4]Smart, N. P. (2003). Cryptography: an introduction (Vol. 3). New York: McGraw-Hill

[5] 2017. [Online]. Available: https://www-uxsup.csx.cam.ac.uk/pub/webmirrors/www.cert.org/incident_notes/IN-98.03.html. [Accessed: 28- Nov- 2017].

[6]"NATO site hacked", *Theregister.co.uk*, 2017. [Online]. Available: http://www.theregister.co.uk/2011/06/24/nato_hack_attack/. [Accessed: 28- Nov- 2017].

[7] John-users - John the Ripper 1.8.0-jumbo-1. (2017). Openwall.com. Retrieved 21 October 2017, from http://www.openwall.com/lists/john-users/2014/12/18/23

[8]Ricardo Rosenfeld y Jerónimo Irazábal, " Computabilidad, complejidad computacional y verificación de programas" Primera edición, 2013 La Plata, Buenos Aires, Argentina

[9]"Message Digest Functions", *Technet.microsoft.com*, 2017. [Online]. Available: https://technet.microsoft.com/en-us/library/cc962033.aspx. [Accessed: 28- Nov- 2017].

[10] http://www.facweb.iitkgp.ernet.in/~sourav/DES.pdf

[11]Gershon Kedem and Yuriko Ishihara. Brute force attack on Unix passwords with SIMD computer. In Proceedings of the 8th conference on USENIX Security Symposium - Volume 8, pages 8–8, Berkeley, CA, USA, 1999. USENIX Association

[12]K. Lab, "¿Qué Es Un Hash Y Cómo Funciona?", *Latam.kaspersky.com*, 2017. [Online]. Available: https://latam.kaspersky.com/blog/que-es-un-hash-y-como-funciona/2806/. [Accessed: 28- Nov- 2017].

[13]B. Byte, "Asi funcionan los Hash Tables", *Bitybyte.github.io*, 2017. [Online]. Available: http://bitybyte.github.io/Hashtables/. [Accessed: 28- Nov- 2017].

[14] CORTEZ, Augusto. Teoría de la complejidad computacional y teoría de la computabilidad. *Universidad Nacional Mayor de San Marcos. Lima, Perú*, 2004.

[15]"IBM Knowledge Center", *Ibm.com*, 2017. [Online]. Available: https://www.ibm.com/support/knowledgecenter/es/SSZSXU_6.2.2.7/com.ibm.tivoli.fim.doc_6227/config/concept/USCSaltnHashExistingFeds.html. [Accessed: 28- Nov- 2017].

[16]c. inglés, "crack Significado en el diccionario Cambridge inglés", *Dictionary.cambridge.org*, 2017. [Online]. Available: https://dictionary.cambridge.org/es/diccionario/ingles/crack. [Accessed: 28- Nov- 2017].

[17] http://dle.rae.es/srv/search?m=30&w=algoritmo

[18] SILVA, John Edward. An overview of cryptographic hash functions and their uses. *GIAC*, 2003.

[19]"Criptoanálisis", *Los diccionarios y las enciclopedias sobre el Académico*, 2017. [Online]. Available: http://www.esacademic.com/dic.nsf/eswiki/79096. [Accessed: 28- Nov- 2017].

[20]"IBM Knowledge Center", *Ibm.com*, 2017. [Online]. Available: https://www.ibm.com/support/knowledgecenter/es/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac55940_.htm. [Accessed: 28- Nov- 2017].

[21]"IBM Research Advances Device Performance for Quantum Computing", *Www-03.ibm.com*, 2017. [Online]. Available: http://www-03.ibm.com/press/us/en/pressrelease/36901.wss. [Accessed: 28- Nov- 2017].

[22]"Computación cuántica", *Es.wikipedia.org*, 2017. [Online]. Available: https://es.wikipedia.org/wiki/Computaci%C3%B3n_cu%C3%A1ntica. [Accessed: 28- Nov- 2017].

[23] Why PWGen? - PWGen. (2017). Pwgen-win.sourceforge.net. Retrieved 21 October 2017, from http://pwgen-win.sourceforge.net/why.html

[24] What can the Oxford English Corpus tell us about language | Oxford Dictionaries. (2017). Oxford Dictionaries | English. Retrieved 21 October 2017, from https://en.oxforddictionaries.com/explore/what-can-corpus-tell-us-about-language